// server.c

2 #include <stdio.h>

3 #include <stdlib.h>

4 #include <string.h>

5 #include <unistd.h>

6 #include <arpa/inet.h>

7 #include <pthread.h>

8

9 #define SERVER\_PORT 8080

10 #define BUFFER\_SIZE 1024

11

12 // Server variables

13 int server\_sock, client\_sock;

14 struct sockaddr\_in server\_addr, client\_addr;

15 socklen\_t client\_len;

16 char buffer[BUFFER\_SIZE];

17

18 // Function prototypes

19 void \*handle\_client\_connection(void \*client\_sock\_desc);

20 void start\_server();

21

22 int main() {

23 start\_server();

24 return 0;

25 }

26

27 void start\_server() {

28 // Create server socket

29 server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

30 if (server\_sock < 0) {

31 perror("Server socket creation failed");

32 exit(1);

33 }

34

35 // Configure server address structure

36 server\_addr.sin\_family = AF\_INET;

37 server\_addr.sin\_addr.s\_addr = INADDR\_ANY; // Accept connections on any IP address

38 server\_addr.sin\_port = htons(SERVER\_PORT);

39

40 // Bind the server socket to the address and port

41 if (bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

42 perror("Bind failed");

43 close(server\_sock);

44 exit(1);

45 }

46

47 // Listen for incoming connections (max 5 clients in the queue)

48 if (listen(server\_sock, 5) < 0) {

49 perror("Listen failed");

50 close(server\_sock);

51 exit(1);

52 }

53 printf("Server listening on port %d...\n", SERVER\_PORT);

54

55 // Accept client connections

56 while (1) {

57 client\_len = sizeof(client\_addr);

58 client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &client\_len);

59 if (client\_sock < 0) {

60 perror("Client connection failed");

61 continue; // Continue listening for new connections

62 }

63

64 printf("Client connected\n");

65

66 // Create a thread for handling the client connection

67 pthread\_t thread\_id;

68 if (pthread\_create(&thread\_id, NULL, handle\_client\_connection, (void \*)&client\_sock) < 0) {

69 perror("Thread creation failed");

70 close(client\_sock);

71 }

72 }

73

74 close(server\_sock);

75 }

76

77 void \*handle\_client\_connection(void \*client\_sock\_desc) {

78 int client\_sock = \*(int \*)client\_sock\_desc;

79 int n;

80

81 // Send greeting message to the client

82 char welcome\_message[] = "Welcome to the server! Type 'exit' to quit.\n";

83 send(client\_sock, welcome\_message, strlen(welcome\_message), 0);

84

85 // Receive client requests

86 while (1) {

87 memset(buffer, 0, BUFFER\_SIZE);

88 n = recv(client\_sock, buffer, BUFFER\_SIZE - 1, 0);

89 if (n <= 0) {

90 printf("Client disconnected or error occurred\n");

91 break;

92 }

93 printf("Received message: %s\n", buffer);

94

95 // Handle client request (e.g., SignUp, Login, etc.)

96 if (strncmp(buffer, "exit", 4) == 0) {

97 send(client\_sock, "Goodbye!\n", 9, 0);

98 break;

99 }

100

101 // Add other client handling logic here as needed (e.g., SignUp, Login, etc.)

102 }

103

104 close(client\_sock);

105 pthread\_exit(NULL);

106 }

107